CONTENTS

1. Introduction 1
2. Hazard Identification 3
3. Hazard Analysis 4
4. Testing 6
5. Hazard Tracking 7
6. Conclusion 9
7. References 9

FIGURES

1. Software Safety Process Flow 2
2. Software vs Hardware Life Cycle 5

ACRONYMS

CDR	Critical Design Review
CSCI	Computer Software Configuration Item
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
HATS	Hazard and Accident Tracking System
H/W	Hardware
I/O	Input and Output
PDR	Preliminary Design Review
PHA	Preliminary Hazard Analysis
SED	Software Engineering Directorate (CECOM)
SRCA	Safety Requirements/Criteria Analysis
SSCA	Software Sneak Circuit Analysis
SSRA	System Safety Risk Assessment
STR	Software Trouble Report
S/W	Software
TDRB	Test Data Review Board
TIR	Test Incident Report

1. Introduction

This report is intended for system and safety engineers who have a working knowledge in the areas of software safety requirements, analysis, design and testing. The purpose of this report is to formalize a process for software safety hazard identification, integration, and tracking for CECOM supported systems. The intent is to identify and evaluate software safety hazards and software development data and integrate these requirements into an existing system safety hazard tracking system. This concept maintains the integrity of a total system hazard tracking process that includes hardware, software, and firmware in accordance with MIL-STD-882C. The report discusses the methodology used to develop additional requirements for software development and software safety that are integrated into a system safety hazard tracking system. Several of the major enhancements include the causes of software safety hazards, the software hazard assessment codes, software versus hardware development life-cycle milestones, software testing phases, priority levels, and areas of software development needing improvement. An attractive feature, the development of a lessons-learned report that is generated based on specific safety hazards, is one of the reasons the requirements are being incorporated.

Background

The hazard identification, integration, and tracking methodology and techniques presented in this report are based on the CECOM Software Safety Guide and MIL-STD-882C. Previous CECOM Safety Office studies have indicated that the category of command and control software programs associated with hazardous hardware functions merits the type of processes presented in this report. Since most of the command and control systems have direct and indirect control (i.e., both control and database generation/manipulation programs) over hazardous functions, there is no advantage in performing extensive analysis on computational/analytical programs unless the output is used to control safety critical functions. Figure 1 is a flowchart of the CECOM software safety processes presented in this report.

The CECOM Safety Office currently enters all our system specific data into the Hazard and Accident Tracking System (HATS). In the future, CECOM could provide software data and lessons-learned information to other agencies, with updates on a periodic basis. This final product greatly enhances our Software System Safety program, and provides a training aid that familiarizes system safety engineers with software safety design principles and development of software safety specifications for new software safety critical applications.

The Report

The following sections present a methodology by which each individual can develop, integrate and tailor his/her own software safety requirements and parameters for inclusion into 2a hazard tracking system: Section 2 describes the initial hazard identification process as the first step in identifying hazardous conditions controlled by the software, and the level of software safety effort/analysis required. Section 3 describes the software safety analysis techniques required to evaluate safety critical errors and the design features needed for corrective action. Section 4 discusses the software testing requirements that are imperative for verification and validation of essential software safety requirements. Section 5 identifies the requirements needed to integrate software safety requirements into an overall system safety HATS.

2. Hazard Identification

Software safety hazard identification should be treated as an extension of the hardware Preliminary Hazard Analysis (PHA). The PHA will ensure that any identified hazard controls related to the hardware system are built into the software design, if possible. The overall objective is to integrate the hardware analysis with the software command and control subsystems (including the operator interface) to generate a complete operational hazard analysis. The two main types of software safety hazards that the CECOM Safety Office has identified and considered in this report are:

1. Command and control software which has direct control of the hardware, and which might cause a hazard (improper command), allow a hazard to propagate (does not detect the occurrence of a hazard), or allow a hazardous condition to go unnoticed by the operator.

2. Programs that generate the data base parameters upon which the command and control software bases its decision.

It is worth noting that, in most cases, the standard software development process and test standards for reliability and quality (i.e., DOD-STD-2167A, 2168, and 498) cover the executive software, I/O software, internal operation of the processor, and self-test software. However, these areas need to be reviewed as part of the qualification testing to verify their capabilities.

During the initial software safety hazard identification process, the system safety engineer should review the PHA, the functional capabilities of the hardware, and, in conjunction with the contractor, evaluate the ways in which the software could create, or allow, a specific problem to occur unnoticed. The integration of the PHA into the software safety analysis would naturally coincide with the hardware Preliminary Design Review (PDR), and emphasize the total system safety concept. Since the software PDR occurs after the hardware PDR, any recommended software design changes identified from the PHA could be incorporated at this time. Adhering to the natural progression of system safety analyses, the Safety Requirements/Criteria Analysis

(SRCA) in accordance with MIL-STD-882C is used to perform the integrated analysis necessary to identify, define and/or refine the software safety design requirements that should be included into the baseline software design documents. At this point in time, the hardware design is established, and the recommendations contained in the SRCA can be incorporated into the software Critical Design Review (CDR). Figure 2 illustrates the hardware versus software life-cycle milestones.

In some cases, the software safety requirements are not known in advance of the PHA. In this case, it is important to include software safety specifications into the software requirement documents. This will ensure traceability of requirements during the testing phase.

During evaluation of the PHA, the PHA should describe the hardware with a total system approach. The PHA should describe the functions allocated to the software and their possible undesired outcome. As part of the PHA, all identified hazards should be mapped to their respective Computer Software Configuration Item (CSCI) for traceability to the CSCI unit level. The results should identify critical functions that will designate critical software tasks. These data, collected from the PHA (i.e., fault or condition, event phase, system effect, hardware and software corrective action, initial software hazard assessment), will be essential for tracking hazards related to software safety.

An initial software system safety checklist can be completed from the data contained in the SRCA, and can be used to help derive requirements associated with the hardware procedural control requirements. To ensure there are no conflicting safety requirements contained in the checklist, the system safety engineer should review all the software requirements. Concurrent engineering between the safety engineer and the software developer/analyst is essential at this point in time. The details of the checklist include program design considerations, failure of the computer processor and any other hardware failures, memory partitioning, complete failures, operator interfaces, H/W-S/W interfaces, illegal entries into critical routines, specific safing actions, timing considerations, anomaly detection, etc. This checklist should be tailored as the software design matures from concept analysis to top level design, and completed after the detailed design review. A comprehensive and tailored checklist should be utilized during the software testing phase.

3. Hazard Analysis

Software System Safety Hazard Analysis should address hazards resulting from deficiencies in the requirement/specification, design, coding, and undesired events. The methodology for extensive software safety analysis includes the above mentioned PHA and checklists, as well as: analyzing hardware and software interfaces, examination of safety critical single and multiple failure sequences, impact of component failures on overall system safety, and evaluation of the design response to safety requirements. The techniques involved in performing software system safety analysis include Fault Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), Software Sneak Circuit Analysis (SSCA), Petri Nets, and design and code walk-throughs. Each of the analyses mentioned have specific advantages and disadvantages in regard to their reasoning, approach, and results. The ultimate goal is to show ways that software failures or errors can contribute to hazards and what software or system hardware monitors, work-arounds or corrective procedures can be used to eliminate or control their effect.

Many of these analyses are traditional hardware safety analyses that can be modified for software safety. The tasks identified in MIL-STD-882C should address specific hazards that have been identified and addressed as a component of the overall system. Tailoring of these tasks is crucial in keeping the cost of the program down. The CECOM Safety Office has directed these labor intensive analyses toward specific safety critical hazardous conditions that exist with the command and control of firing and lasing systems.

The results from the SRCA and the Software Safety Design Analysis should be addressed at the software Critical Design Review. Specific software safety design recommendations that the CECOM Safety Office has incorporated into software safety critical programs include watchdog timers, two-fault rule for enabling entrance into critical routines, independent interrupt routines, timing constraints, fail-safe recovery, and memory allocation.

After the software system safety analysis is completed, any software safety design recommendations that become part of the software requirement design documents should be incorporated into the configuration control process for verification, validation and tracking purposes.

4. Testing

Software system safety testing verifies that the safety requirements (i.e., inhibits, traps, interlocks) have been correctly implemented. Software safety testing also verifies that the software functions safely within its intended environment. Many times software safety testing reveals that software safety requirements are in direct conflict with military operational doctrine. An example of this is firing into no-fire zones or boundary areas. A commander, based on the threat, can make a tactical decision to fire in the vicinity of friendlies, and violate safety design features. If this is the case, it is very important that a thorough review of the intended operation of the system be conducted with the user community (i.e., Training and Doctrine Command). Usually a safety critical function will not be overridden by operational doctrine. However, if the user decides that there is a work-around or operational doctrine that overrides the incorporation of a software safety design requirement, then a System Safety Risk Assessment (SSRA) is processed.

Software system safety test requirements are derived from the PHA, Checklists, and any other system safety hazard analyses. Many requirements are also gained from the software design documentation and test plans. As part of the software system safety test analysis, safety engineers should evaluate safety-related test descriptions, procedures, cases, and qualification criteria for areas needing software safety-critical test input. The next step should be the identification of specific safety tests that will be required for each software safety-critical module and program. At this point in time, the safety engineer should keep a log of actual safety-related tests that are carried out, with the details and results of the testing. For CECOM supported systems, this will aid in the development of software safety suitability for release statements which are needed to field the software. The software safety suitability for release statement is a comprehensive evaluation of the safety of the system, prepared by the CECOM Safety Office. All other directorates within the command, including the Software Engineering Directorate (SED), are required to prepare their suitability for release statements.

Software safety testing may include, but is not limited to, computer software unit level testing, hardware-software operator interface testing, stress testing, go-no-go path testing, regression testing, and failure mode testing. It should be noted that any patches made to the baseline version of software safety critical systems should undergo complete regression testing for fielding as a complete package.

During the scoring of specific safety-critical tests, a safety engineer should be present to evaluate their impact and determine/assign a priority level to the Software Trouble Report (STR) or Test Incident Report (TIR). Priority levels rank from the highest (1) to the lowest (5) and assess the software error. A priority 1 classification is defined as a software problem that jeopardizes personnel safety, and a priority 2 classification specifies that the software problem adversely affects the accomplishment of an operational or mission essential capability, and no alternative work-around solution is known. For CECOM supported systems any safety critical STR is given a priority 1 or 2 and must be corrected. CECOM does not classify any safety critical errors with a priority of 3 or less. The Test Data Review Board (TDRB) is an excellent mechanism for safety engineers to review, discuss, and evaluate safety-critical STR's. A thorough review of the STR will reveal the actual software requirement specification that initiated the test case and a trace can be done to locate the source code where the software error occurred.

5. <u>Hazard Tracking</u>

In order to comprehensively track software safety-related hazards, the CECOM Safety hazard tracking process was modified to incorporate the total system safety approach as delineated in MIL-STD-882C. MIL-STD-882C has integrated software safety requirements into the system safety tasks, and has incorporated a process for software safety hazard assessment, including a software safety hazard criticality matrix.

All the existing system safety fields can be used for systems that include software safety requirements. The CECOM Safety Office has designed the interface to be system oriented and many menus are context sensitive. If safety-critical software is not part of your system, then many specific software safety-related fields will not be invoked.

Several of the processes and analyses discussed above include critical information that is incorporated into the CECOM HATS. This information is integrated into system safety data/hazards and follows the natural life-cycle of major system development, where applicable. The tendency to separate hardware and software efforts would not lend itself to the total system approach. Furthermore, the CECOM Safety Office has designed the HATS fields/data to act as a lessons-learned repository, where specific safety reports can be generated.

The additional software safety information/data that are imperative for an extensive hazard tracking system, and are included in the CECOM HATS, include the following:

System Safety Data Record:

- Software version.
- Software requirements included in the contract.
- Software design language.
- Type of standards that the software is designed to.
- What design guidelines are being used.
- Was a software safety checklist required.
- What type of software safety analysis was performed.
- Software PDR and CDR schedules.
- Software system test types.
- Software safety suitability for release statement.

System Safety Hazard Record:

- Added the "software version" as part of the system components.
- Added "hold" and "monitor" to the hazard status.
- Added "software safety analysis" and "software testing" to the "event that first identified the hazard."
- Added "priority level assigned," if software testing identified the hazard.
- Added "software" to hazard type.
- Added "system or operation affected."
- Added the "software hazard assessment categories."
- Added "areas needing improvement."

It should be noted that several categories are included for the development of a system safety lessons-learned repository, from which specific hazard reports can be generated. Lessons learned are extremely important not only for developing system safety requirements, but are crucial to the certification of software safety on any program. The CECOM Safety Office realizes that lessons learned must augment training.

6. Conclusion

The incorporation of Software System Safety requirements into the system safety hazard tracking system teaches system safety engineers the processes, specifications, necessary information, and hazards involved with a Software System Safety program. This also allows safety engineers to obtain baseline data for new systems and establish a standardization of requirements between contractors and Government. The ability to have a centralized source of software safety information that includes hazard controls, risk assessments, and report generation is a valuable tool. The CECOM HATS assists engineers in developing requirements and establishing an integrated software safety program. In the future, the CECOM Safety Office plans on researching the software hazard risk assessment process as it relates to software priority classifications and problem reporting. In conclusion, this is a new discipline which is still evolving and we must follow the logical progression of an integrated system safety effort.

7. References

1. 12th International System Safety Conference Proceedings, "Identification, Integration, and Tracking of Software System Safety Requirements," 6 July 94.

2. CECOM Safety Technical Report, TR-92-2, "Software System Safety Guide," DTIC AD No. A250321, May 92.

3. CECOM Regulation, CECOM Reg 385-21, "Software System Safety," 19 Feb 91.

4. DOD-STD-2167A, "Defense System Software Development," 29 Feb 88.

5. DOD-STD-2168, "Defense System Software Quality Program," 29 April 88.

6. MIL-STD-498 (Draft), "Software Development and Documentation," 30 March 94.

7. MIL-STD-882C, "System Safety Program Requirements," 19 Jan 93.